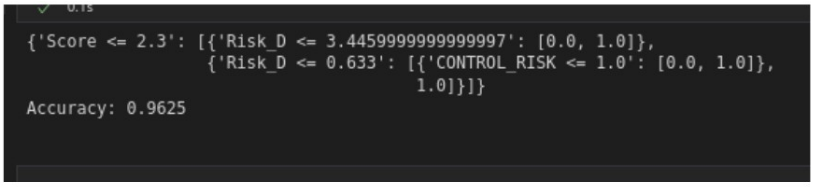
Gebze Technical University

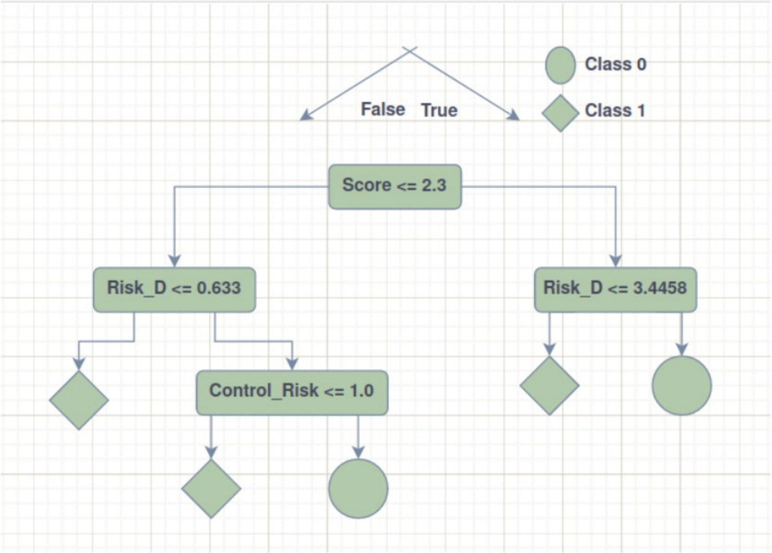
CSE455 Machine Learning Course
Homework 2

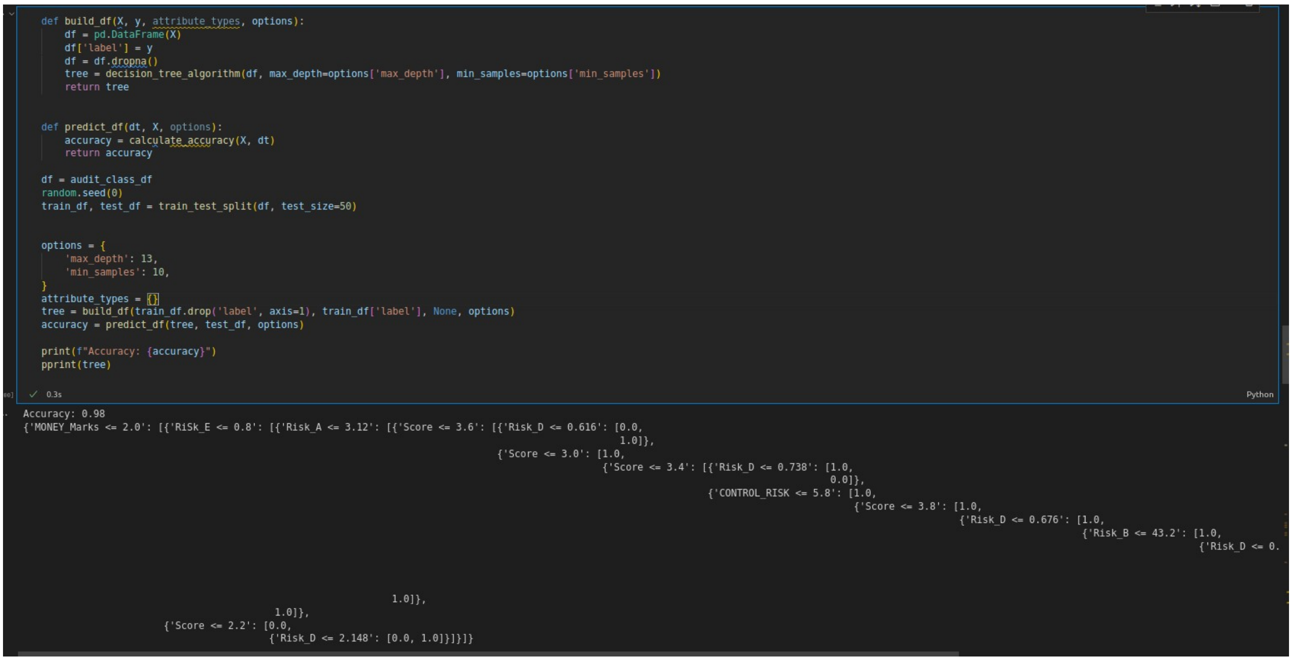
Muhammed Bedir ULUCAY
1901042697

Classification Decision Tree Output Results:



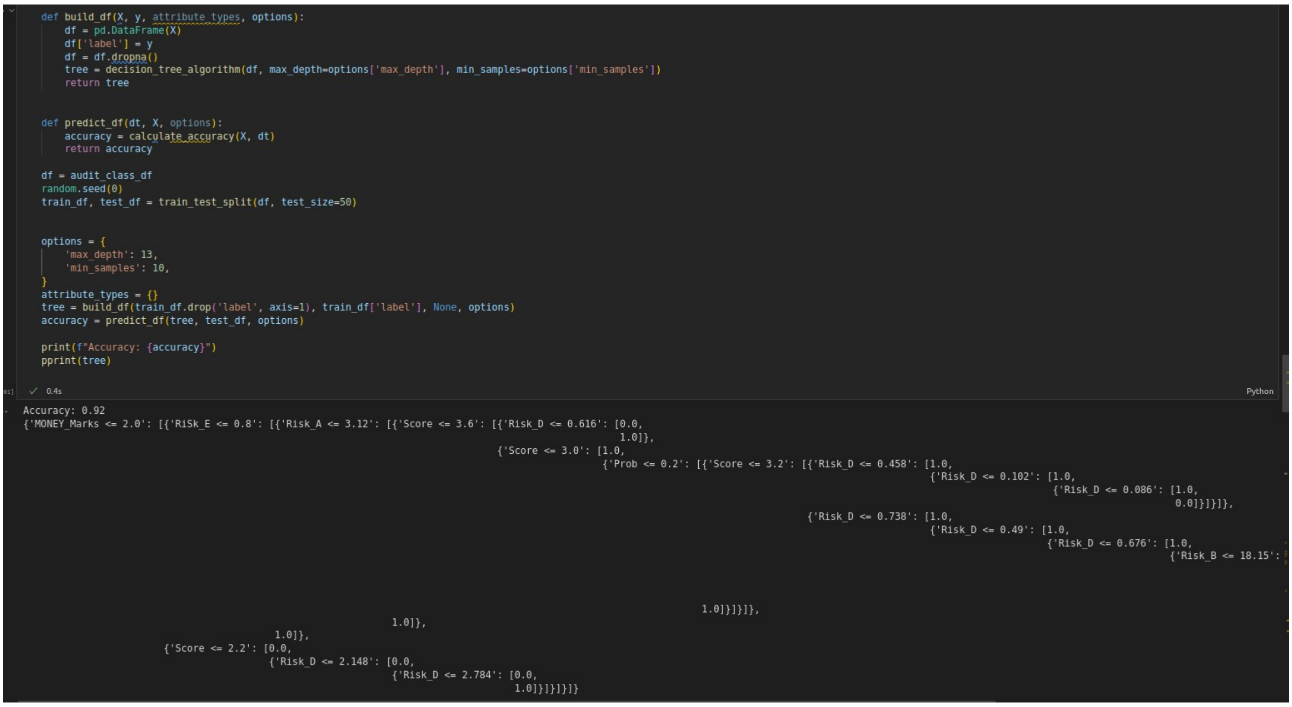
```
{'Score <= 2.3': [{'Risk_D <= 3.4459999999999997': [0.0, 1.0]},
                  {'Risk_D <= 0.633': [{'CONTROL_RISK <= 1.0': [0.0, 1.0]},
                                       1.0]}]}
Accuracy: 0.9625
```





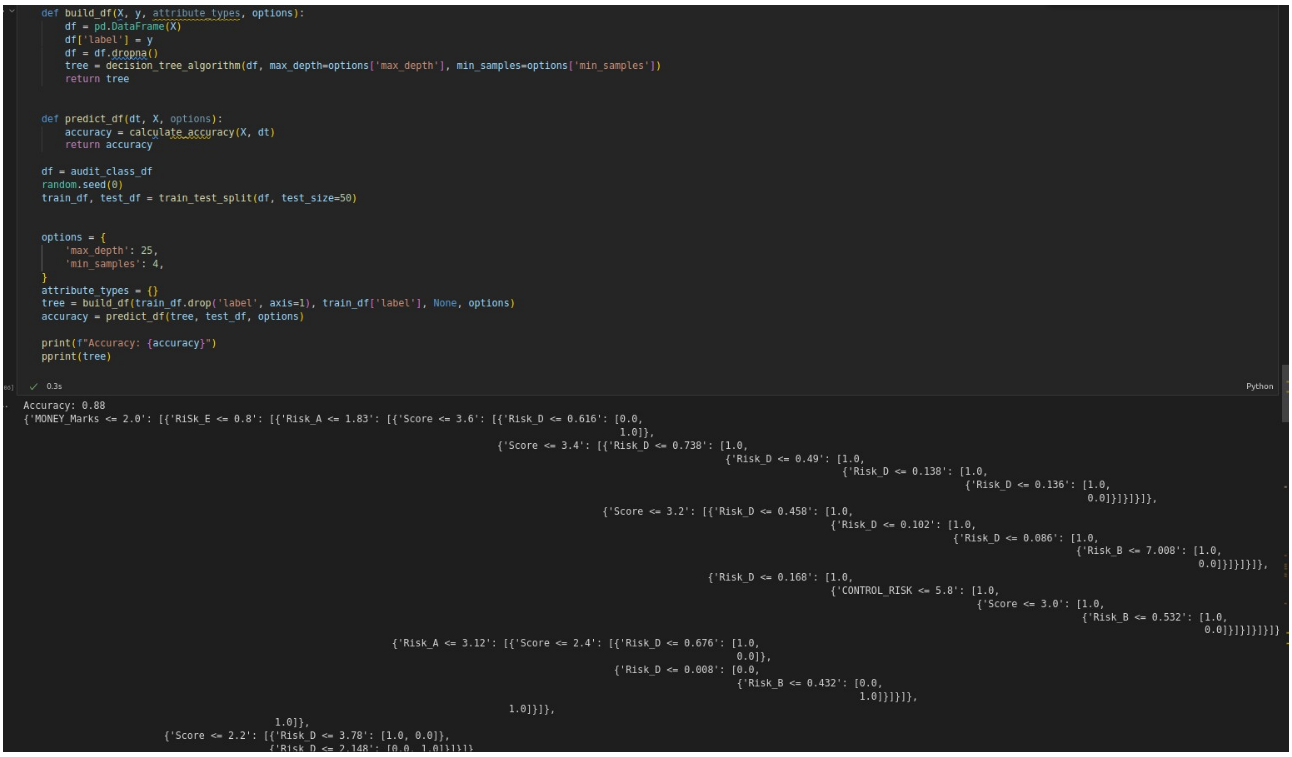
```python
def build_df(X, y, attribute_types, options):
    df = pd.DataFrame(X)
    df['label'] = y
    df = df.dropna()
    tree = decision_tree_algorithm(df, max_depth=options['max_depth'], min_samples=options['min_samples'])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

df = audit_class_df
random.seed(0)
train_df, test_df = train_test_split(df, test_size=50)


options = {
    'max_depth': 13,
    'min_samples': 10,
}
attribute_types = {}
tree = build_df(train_df.drop('label', axis=1), train_df['label'], None, options)
accuracy = predict_df(tree, test_df, options)

print(f"Accuracy: {accuracy}")
pprint(tree)
```

```
Accuracy: 0.98
{'MONEY_Marks <= 2.0': [{'RiSk_E <= 0.8': [{'Risk_A <= 3.12': [{'Score <= 3.6': [{'Risk_D <= 0.616': [0.0,
                                                                                                      1.0]},
                                                                                 {'Score <= 3.0': [1.0,
                                                                                                   {'Score <= 3.4': [{'Risk_D <= 0.738': [1.0,
                                                                                                                                          0.0]},
                                                                                                                     {'CONTROL_RISK <= 5.8': [1.0,
                                                                                                                                              {'Score <= 3.8': [1.0,
                                                                                                                                                                {'Risk_D <= 0.676': [1.0,
                                                                                                                                                                                     {'Risk_B <= 43.2': [1.0,
                                                                                                                                                                                                         {'Risk_D <= 0.



                                                               1.0]},
                                           1.0]},
                        {'Score <= 2.2': [0.0,
                                          {'Risk_D <= 2.148': [0.0, 1.0]}]}]}
```



```python
def build_df(X, y, attribute_types, options):
    df = pd.DataFrame(X)
    df['label'] = y
    df = df.dropna()
    tree = decision_tree_algorithm(df, max_depth=options['max_depth'], min_samples=options['min_samples'])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

df = audit_class_df
random.seed(0)
train_df, test_df = train_test_split(df, test_size=50)


options = {
    'max_depth': 13,
    'min_samples': 10,
}
attribute_types = {}
tree = build_df(train_df.drop('label', axis=1), train_df['label'], None, options)
accuracy = predict_df(tree, test_df, options)

print(f"Accuracy: {accuracy}")
pprint(tree)
```

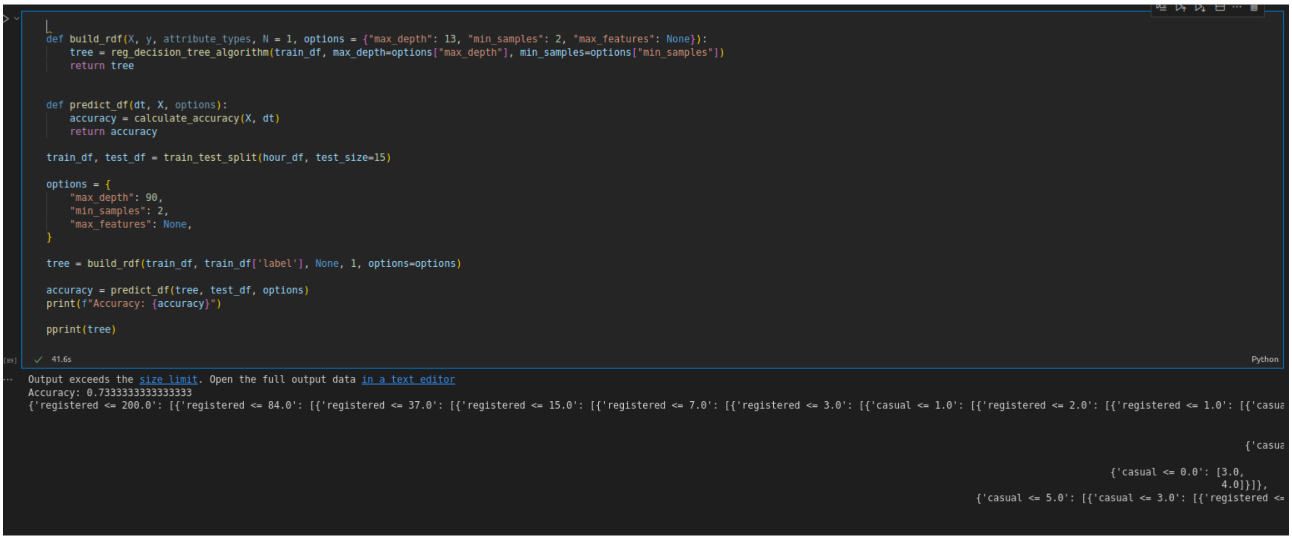
```
Accuracy: 0.92
{'MONEY_Marks <= 2.0': [{'RiSk_E <= 0.8': [{'Risk_A <= 3.12': [{'Score <= 3.6': [{'Risk_D <= 0.616': [0.0,
                                                                                                      1.0]},
                                                                                 {'Score <= 3.0': [1.0,
                                                                                                   {'Prob <= 0.2': [{'Score <= 3.2': [{'Risk_D <= 0.458': [1.0,
                                                                                                                                                           {'Risk_D <= 0.102': [1.0,
                                                                                                                                                                                {'Risk_D <= 0.086': [1.0,
                                                                                                                                                                                                     0.0]}]}]},
                                                                                                                                      {'Risk_D <= 0.738': [1.0,
                                                                                                                                                           {'Risk_D <= 0.49': [1.0,
                                                                                                                                                                               {'Risk_D <= 0.676': [1.0,
                                                                                                                                                                                                    {'Risk_B <= 18.15':



                                                                                                                    1.0]}]}]},
                                                               1.0]},
                                           1.0]},
                        {'Score <= 2.2': [0.0,
                                          {'Risk_D <= 2.148': [0.0,
                                                               {'Risk_D <= 2.784': [0.0,
                                                                                    1.0]}]}]}]}
```



```python
def build_df(X, y, attribute_types, options):
    df = pd.DataFrame(X)
    df['label'] = y
    df = df.dropna()
    tree = decision_tree_algorithm(df, max_depth=options['max_depth'], min_samples=options['min_samples'])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

df = audit_class_df
random.seed(0)
train_df, test_df = train_test_split(df, test_size=50)


options = {
    'max_depth': 25,
    'min_samples': 4,
}
attribute_types = {}
tree = build_df(train_df.drop('label', axis=1), train_df['label'], None, options)
accuracy = predict_df(tree, test_df, options)

print(f"Accuracy: {accuracy}")
pprint(tree)
```

```
Accuracy: 0.88
{'MONEY_Marks <= 2.0': [{'RiSk_E <= 0.8': [{'Risk_A <= 1.83': [{'Score <= 3.6': [{'Risk_D <= 0.616': [0.0,
                                                                                                      1.0]},
                                                                                 {'Score <= 3.4': [{'Risk_D <= 0.738': [1.0,
                                                                                                                        {'Risk_D <= 0.49': [1.0,
                                                                                                                                            {'Risk_D <= 0.138': [1.0,
                                                                                                                                                                 {'Risk_D <= 0.136': [1.0,
                                                                                                                                                                                      0.0]}]}]}]},
                                                                                                   {'Score <= 3.2': [{'Risk_D <= 0.458': [1.0,
                                                                                                                                          {'Risk_D <= 0.102': [1.0,
                                                                                                                                                               {'Risk_D <= 0.086': [1.0,
                                                                                                                                                                                    {'Risk_B <= 7.008': [1.0,
                                                                                                                                                                                                         0.0]}]}]}]},
                                                                                                                     {'Risk_D <= 0.168': [1.0,
                                                                                                                                          {'CONTROL_RISK <= 5.8': [1.0,
                                                                                                                                                                   {'Score <= 3.0': [1.0,
                                                                                                                                                                                     {'Risk_B <= 0.532': [1.0,
                                                                                                                                                                                                          0.0]}]}]}]}]}]}
                                                               {'Risk_A <= 3.12': [{'Score <= 2.4': [{'Risk_D <= 0.676': [1.0,
                                                                                                                          0.0]},
                                                                                                     {'Risk_D <= 0.008': [0.0,
                                                                                                                          {'Risk_B <= 0.432': [0.0,
                                                                                                                                               1.0]}]}]},
                                                                                   1.0]}]},
                                           1.0]},
                        {'Score <= 2.2': [{'Risk_D <= 3.78': [1.0, 0.0]},
                                          {'Risk_D <= 2.148': [0.0, 1.0]}]}]}
```

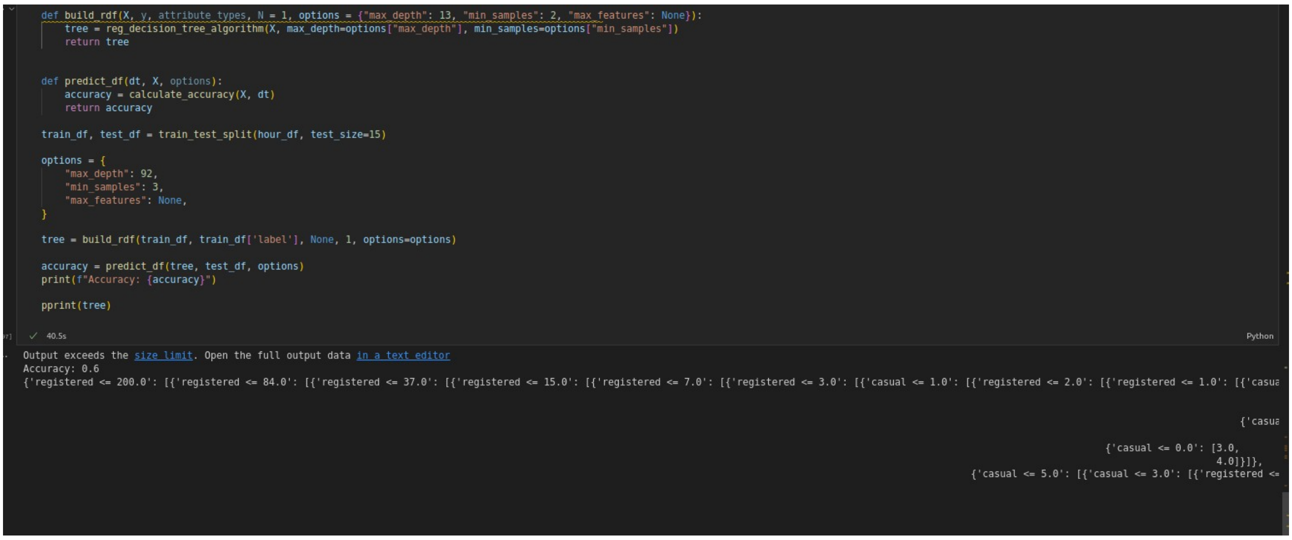
Regression Decision Tree Output Results:



```
def build_rdf(X, y, attribute_types, N = 1, options = {"max_depth": 13, "min_samples": 2, "max_features": None}):
    tree = reg_decision_tree_algorithm(train_df, max_depth=options["max_depth"], min_samples=options["min_samples"])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

train_df, test_df = train_test_split(hour_df, test_size=15)

options = {
    "max_depth": 90,
    "min_samples": 2,
    "max_features": None,
}

tree = build_rdf(train_df, train_df['label'], None, 1, options=options)

accuracy = predict_df(tree, test_df, options)
print(f"Accuracy: {accuracy}")

pprint(tree)
```

[89] ✓ 41.6s Python

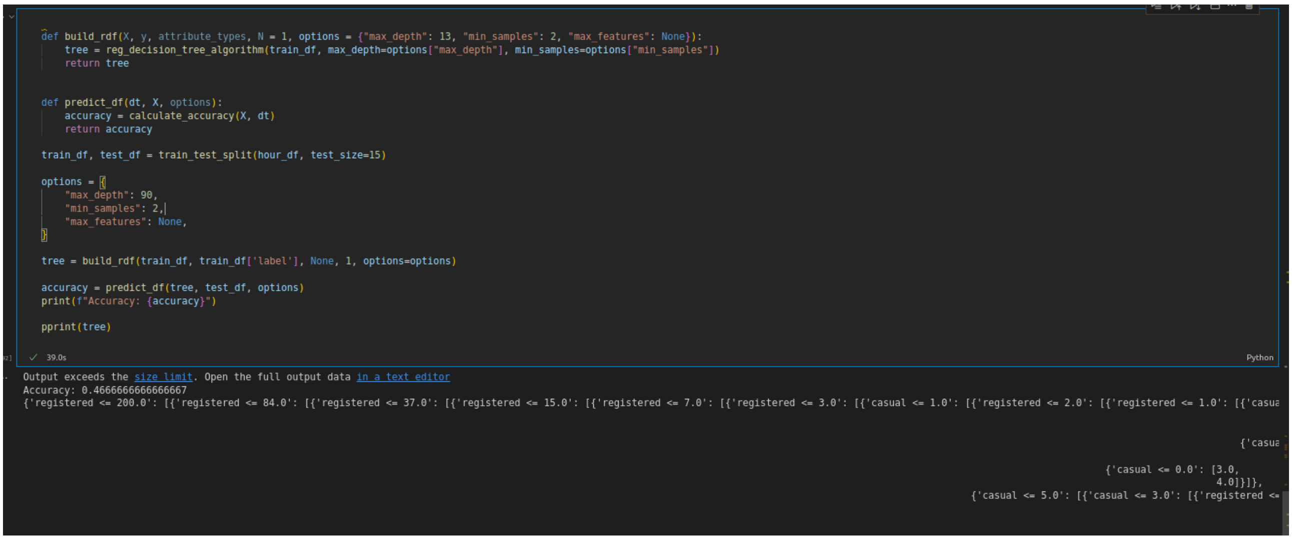
```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy: 0.7333333333333333
{'registered <= 200.0': [{'registered <= 84.0': [{'registered <= 37.0': [{'registered <= 15.0': [{'registered <= 7.0': [{'registered <= 3.0': [{'casual <= 1.0': [{'registered <= 2.0': [{'registered <= 1.0': [{'casua

                                                                                                                                                                                                               {'casua

                                                                                                                                                                                        {'casual <= 0.0': [3.0,
                                                                                                                                                                                                           4.0]}]},
                                                                                                                                                                  {'casual <= 5.0': [{'casual <= 3.0': [{'registered <=
```



```
def build_rdf(X, y, attribute_types, N = 1, options = {"max_depth": 13, "min_samples": 2, "max_features": None}):
    tree = reg_decision_tree_algorithm(X, max_depth=options["max_depth"], min_samples=options["min_samples"])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

train_df, test_df = train_test_split(hour_df, test_size=15)

options = {
    "max_depth": 92,
    "min_samples": 3,
    "max_features": None,
}

tree = build_rdf(train_df, train_df['label'], None, 1, options=options)

accuracy = predict_df(tree, test_df, options)
print(f"Accuracy: {accuracy}")

pprint(tree)
```

✓ 40.5s Python

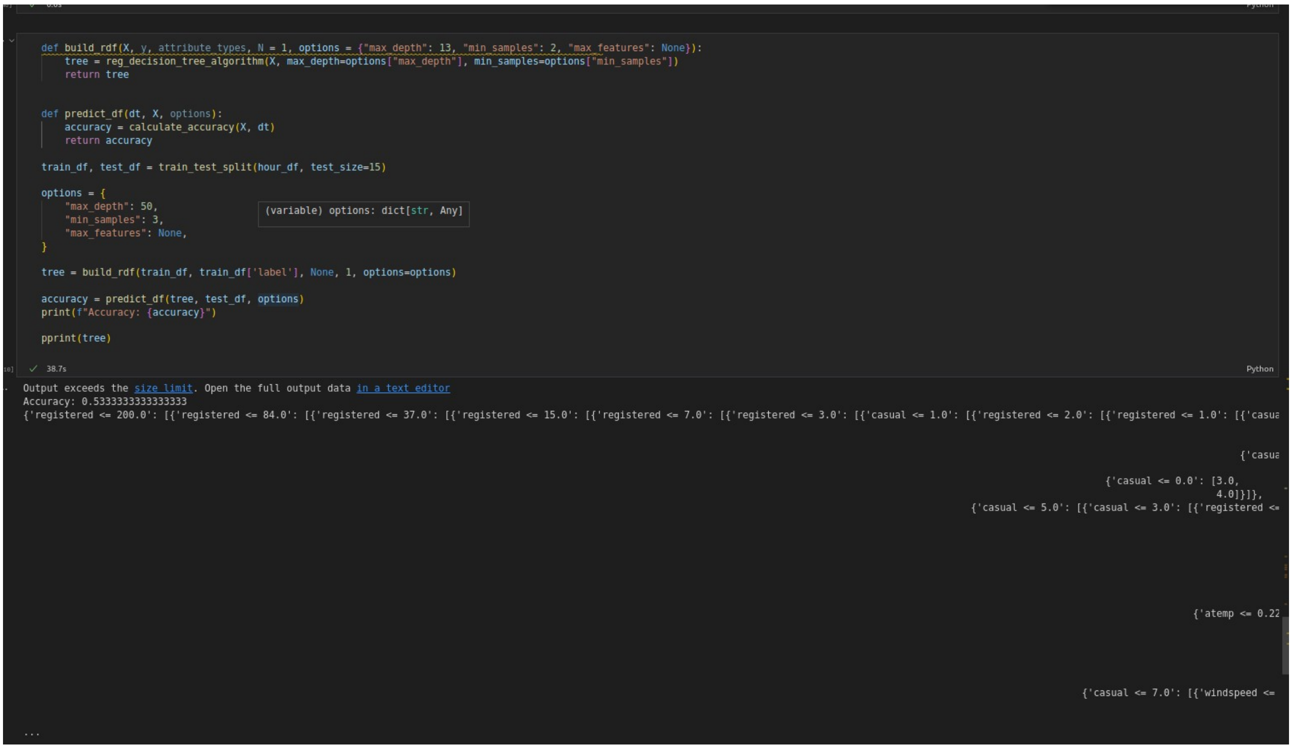
```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy: 0.6
{'registered <= 200.0': [{'registered <= 84.0': [{'registered <= 37.0': [{'registered <= 15.0': [{'registered <= 7.0': [{'registered <= 3.0': [{'casual <= 1.0': [{'registered <= 2.0': [{'registered <= 1.0': [{'casua

                                                                                                                                                                                                               {'casua

                                                                                                                                                                                        {'casual <= 0.0': [3.0,
                                                                                                                                                                                                           4.0]}]},
                                                                                                                                                                  {'casual <= 5.0': [{'casual <= 3.0': [{'registered <=
```



```
def build_rdf(X, y, attribute_types, N = 1, options = {"max_depth": 13, "min_samples": 2, "max_features": None}):
    tree = reg_decision_tree_algorithm(train_df, max_depth=options["max_depth"], min_samples=options["min_samples"])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

train_df, test_df = train_test_split(hour_df, test_size=15)

options = {
    "max_depth": 90,
    "min_samples": 2,
    "max_features": None,
}

tree = build_rdf(train_df, train_df['label'], None, 1, options=options)

accuracy = predict_df(tree, test_df, options)
print(f"Accuracy: {accuracy}")

pprint(tree)
```

✓ 39.0s Python

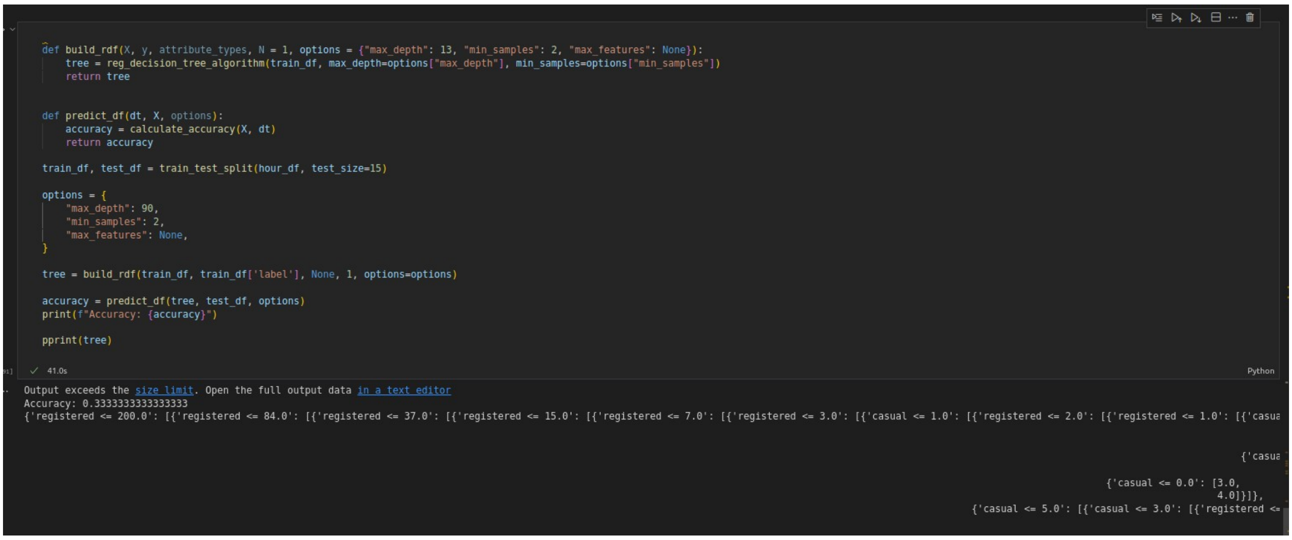
```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy: 0.4666666666666667
{'registered <= 200.0': [{'registered <= 84.0': [{'registered <= 37.0': [{'registered <= 15.0': [{'registered <= 7.0': [{'registered <= 3.0': [{'casual <= 1.0': [{'registered <= 2.0': [{'registered <= 1.0': [{'casua

                                                                                                                                                                                                               {'casua

                                                                                                                                                                                        {'casual <= 0.0': [3.0,
                                                                                                                                                                                                           4.0]}]},
                                                                                                                                                                  {'casual <= 5.0': [{'casual <= 3.0': [{'registered <=
```



```python
def build_rdf(X, y, attribute_types, N = 1, options = {"max_depth": 13, "min_samples": 2, "max_features": None}):
    tree = reg_decision_tree_algorithm(X, max_depth=options["max_depth"], min_samples=options["min_samples"])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

train_df, test_df = train_test_split(hour_df, test_size=15)

options = {
    "max_depth": 50,
    "min_samples": 3,
    "max_features": None,
}

tree = build_rdf(train_df, train_df['label'], None, 1, options=options)

accuracy = predict_df(tree, test_df, options)
print(f"Accuracy: {accuracy}")

pprint(tree)
```

(variable) options: dict[str, Any]

```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy: 0.5333333333333333
{'registered <= 200.0': [{'registered <= 84.0': [{'registered <= 37.0': [{'registered <= 15.0': [{'registered <= 7.0': [{'registered <= 3.0': [{'casual <= 1.0': [{'registered <= 2.0': [{'registered <= 1.0': [{'casua


                                                                                                                                                                                                               {'casua

                                                                                                                                                                                        {'casual <= 0.0': [3.0,
                                                                                                                                                                                                           4.0]}]},
                                                                                                                                                                 {'casual <= 5.0': [{'casual <= 3.0': [{'registered <=







                                                                                                                                                                                                       {'atemp <= 0.22





                                                                                                                                                                                    {'casual <= 7.0': [{'windspeed <=


...
```



```python
def build_rdf(X, y, attribute_types, N = 1, options = {"max_depth": 13, "min_samples": 2, "max_features": None}):
    tree = reg_decision_tree_algorithm(train_df, max_depth=options["max_depth"], min_samples=options["min_samples"])
    return tree


def predict_df(dt, X, options):
    accuracy = calculate_accuracy(X, dt)
    return accuracy

train_df, test_df = train_test_split(hour_df, test_size=15)

options = {
    "max_depth": 90,
    "min_samples": 2,
    "max_features": None,
}

tree = build_rdf(train_df, train_df['label'], None, 1, options=options)

accuracy = predict_df(tree, test_df, options)
print(f"Accuracy: {accuracy}")

pprint(tree)
```

```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy: 0.3333333333333333
{'registered <= 200.0': [{'registered <= 84.0': [{'registered <= 37.0': [{'registered <= 15.0': [{'registered <= 7.0': [{'registered <= 3.0': [{'casual <= 1.0': [{'registered <= 2.0': [{'registered <= 1.0': [{'casua


                                                                                                                                                                                                               {'casua

                                                                                                                                                                                        {'casual <= 0.0': [3.0,
                                                                                                                                                                                                           4.0]}]},
                                                                                                                                                                 {'casual <= 5.0': [{'casual <= 3.0': [{'registered <=
```